Deployment Automation with BLITZ

Brian Dougherty, Jules White, Jaiganesh Balasubramanian, Chris Thompson, Douglas C. Schmidt
Vanderbilt University USA

Abstract

Minimizing the computing infrastructure (such as processors) in a distributed real-time embedded (DRE) system deployment helps reduce system size, weight, power consumption, and cost. To support software components and applications on the computing infrastructure, the hardware must provide enough processors to ensure that all applications can be scheduled without missing real-time deadlines. In addition to ensuring scheduling constraints, sufficient resources (such as memory) must be available to the software. It is hard to identify the best way(s) of deploying software components on hardware processors to minimize computing infrastructure and meet complex DRE
constraints.
This poster provides the following contributions to the study of the deployment of software components to hardware in DRE systems: (1) we present an algorithmic deployment technique that minimizes the required number of processors, while adhering to real-time scheduling, resource, and co-location constraints, (2) we show how this technique can be augmented with a harmonic period heuristic to further reduce the number of required processors, and (3) we present empirical data from applying three different deployment algorithms for processor minimization to a flight avionics DRE system.

Solution Approach: BLITZ

What is Blitz?

The Binpacking LocatIon Technique for processor minimiZation} (BLITZ) is a first-fit decreasing binpacking algorithm we developed to (1) assign processor utilization values that ensure schedulability if not exceeded and (2) enhance existing techniques by ensuring that multiple resource and co-location constraints are simultaneously honored.

How Does BLITZ Work?

1.)First Fit Decreasing Bin-Packing

In BLITZ, each processor is modeled as a bin and each independent component or co-located group of components is modeled as an item. Each bin has a dimension corresponding to the available CPU utilization. Each item has a dimension that represents the CPU utilization it requires, as well as a dimension corresponding to each resource, such as memory, that it consumes. Each bin's size dimension corresponding to available CPU utilization is initialized 100%. The resource dimensions are set to the amount of each resource that the processor offers.
To pack the items, they are first sorted in decreasing order of utilization. Next, BLITZ attempts to place the first item in the first bin. If the placement of the item does not exceed the size of the bin (available resources and utilization) of the bin (processor), the item is placed in the bin. The dimensions of the items are then subtracted from the dimensions of the bin to reflect the addition. If the item does not fit, BLITZ attempts to insert the item into the next bin. This step is repeated until all items are packed into bins or no bin exists that can contain the item.

2.)Utilization Bounds

Conventional bin-packing algorithms assume that each bin has a static series of dimensions corresponding to available resources. For example, the amount of RAM provided by the processor is constant. Applying conventional bin-packing algorithms to software component deployment is a challenge since it is hard to set a static bin dimension that guarantees the components are schedulable. Scheduling can only be modeled with a constant bin dimension of utilization if a worst-case scheduling of the system is assumed. Liu-Layland[1] have shown that a fixed bin dimension of 69.4\% will guarantee schedulability but in many cases, tasks can have a higher utilization and still be schedulable.
The Liu-Layland equation states that the maximum processor utilization that guarantees schedulability is equal to $2^{(1/x)}$-1, where x is the total number of components allocated to the processor. With BLITZ, each bin has a scheduling dimension that is determined by the Liu-Layland equation and the number of components currently assigned to the bin. Each item will represent at least one but possibly multiple co-located components. Each time an item is assigned to a bin, BLITZ uses the Liu-Layland formula to dynamically resize the bin's scheduling dimension according to the number of components held by the items in the bin.
If the the frequency of execution, or periodicity, of the components' execution requirements is known, higher processor utilization above the Liu-Layland bound is also possible. Components with harmonic periods (e.g. periods that can be repeatedly doubled or halved to equal each other) can be allocated to the same processor with schedulability ensured, as long as the total utilization is less than or equal to 100%.

3.)Co-location Constraints

To allow for component co-location constraints, BLITZ groups components that require co-location into a single item. Each item has utilization and resource consumption equal to that of the component(s) it represents. Each item remembers the components associated with it. The Liu-Layland and harmonic calculations are performed on the individual components associated with the items in a bin and not each item as a whole.

Empirical Results